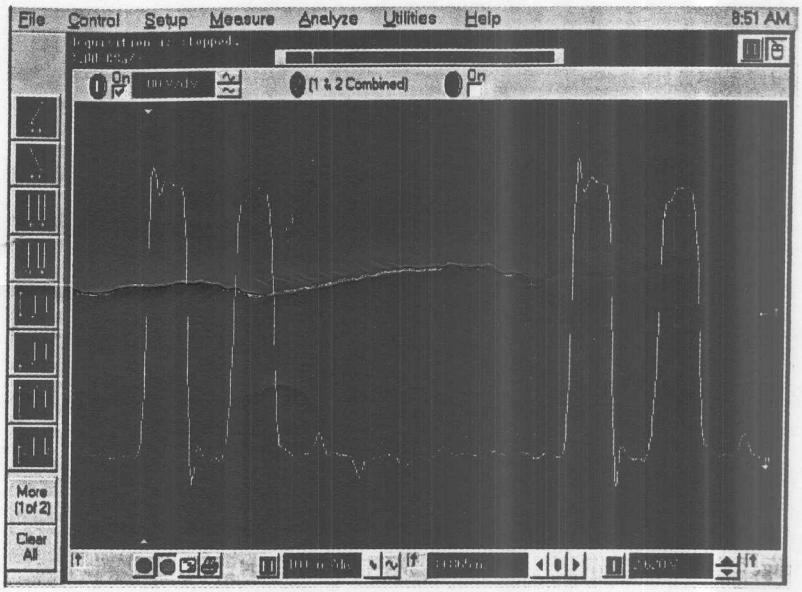
cilloscope measures is virtually all oscilloscope jitter. The loop allowed a generator with 19-psec rms jitter to calibrate an oscilloscope having 3.8-nsec jitter.

Figure 2



The transmission-line delay loop creates a delayed signal with near-zero jitter for calibrating the oscilloscope.

The delay loop has some jitter, created by the conversion of amplitude noise to jitter. Without the loop, generator-amplitude noise causes the leading edge of each pulse to cross the oscilloscope's trip point either early or late. In this way, amplitude noise translates to jitter. The following formula gives jitter versus amplitude noise:

$$\text{JITTER} = \frac{\text{AMPLITUDE NOISE}}{dV/dt}.$$

An ideal loop would produce a delayed pulse that is identical to the first pulse. Amplitude noise on the delayed pulse would be identical to that on the first pulse; thus, jitter attributable to the noise would cancel out. Because of signal loss in a real loop, the delayed pulse is not identical to the first pulse. Therefore, the amplitude noise of the delayed pulse will be less that that on the first pulse, and a conversion of amplitude noise to jitter will occur. The following formula gives amplitude noise versus jitter in the loop:

$$\text{LOOP JITTER} = \frac{\text{AMPLITUDE NOISE}}{dV/dt} \times (\text{SIGNAL LOSS}).$$

The pulse generator used to calibrate the oscilloscopes exhibits 250 μV of rms noise and a dV/dt of 0.35V/nsec. The signal loss on the leading edge of the delayed pulse is 0.2V. The amplitude-noise-to-jitter conversion is thus:

$$\text{LOOP JITTER} = \frac{250\,\mu V}{0.35V/nSEC} \times (0.20) = 143\ fSEC.$$

The loop jitter of 145 fsec is so far below the jitter noise floor of the oscilloscopes under calibration that you can consider the delay loop as a zero-jitter source. Because digital-oscilloscope jitter is a function of the timebase setting and the amount of ADC dynamic range you use, you should calibrate the scope with a loop delay and amplitude that match the signal to the actual system or device under test.

REFERENCE

1. Adler, Joe, "Jitter in clock sources," Application Note, Vectron International.

Is this the best Design Idea in this issue? Vote at www.ednmag.com.

Excel offers painless LCD initialization

Alberto Bitti, Eptar, Lugo, Italy

TO DISPLAY A FONT or a symbol on an LCD, you need to convert the desired character into numerical data. Creating the data for an entire font set requires specialized tools; even with these tools, the task can be daunting. Alternatively, you can build a font calculator using an Excel spreadsheet. This technique takes advantage of the tabular nature of a spreadsheet to automatically create the required initialization code. The example in **Figure 1** applies to displays arranged in blocks comprised of eight pixels by eight pixels. You can easily adapt it to any other arrangement, such as the seven-by-five-pixel format used in the eight customizable characters found in most alphanumeric displays. Cells inside the yellow area represent the symbol "pixels." You draw the desired font character or symbol using a bold character such as "#." The formula in **Figure 2** is all that

font calculator.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1												{				
2				#	#	#	#				60	,				
3			#	#			#	#			102	,				
4							#	#			6	,				
5						#	#				12	,				
6					#	#					24	,				
7				#	#						48	,				
8			#	#	#	#	#	#			126	,				
9											0					
10												}				
11																

Figure 1 **You edit the graphical shape (yellow). A simple formula builds the initialization code (blue), which is ready to paste in your application.**

you need to obtain the initialization data. For each of the eight pixels in a row, the formula tests whether the pixel is blank (LEN is zero); otherwise, it adds the pixel's "weight" (1, 2, 4, 8, 16, 32, 64, or 128) to the result.

You can prepare the spreadsheet in minutes. Type the formula in the first position (K2), then copy it to all eight rows in a symbol (the blue area from K3 to K9). Complete it with the separators for your language of choice (commas and parentheses for C). To build an entire character set, copy the whole block as many times as necessary. To edit the fonts, place the cursor over the cells and type "#" (or any other character), and use "Canc" to delete. After editing, select all of the columns K and L; then cut and paste the code into your application. Besides saving you money, this technique is convenient and flexible. You can adapt it in minutes to any language (useful when you switch between assembly dialects). Moreover, the method accommodates useful additions, such as inserting #define KEY_ICON to name a particular data set, to suit your application's requirements.

Figure 2

```
= IF( LEN(B2);128; 0 ) +
  IF( LEN(C2); 64; 0 ) +
  IF( LEN(D2); 32; 0 ) +
  IF( LEN(E2); 16; 0 ) +
  IF( LEN(F2);  8; 0 ) +
  IF( LEN(G2);  4; 0 ) +
  IF( LEN(H2);  2; 0 ) +
  IF( LEN(I2);  1; 0 )
```

The formula (which, in this example, shows cell K2) adds pixel "weights" to obtain initialization values.

Is this the best Design Idea in this issue? Vote at www.ednmag.com.

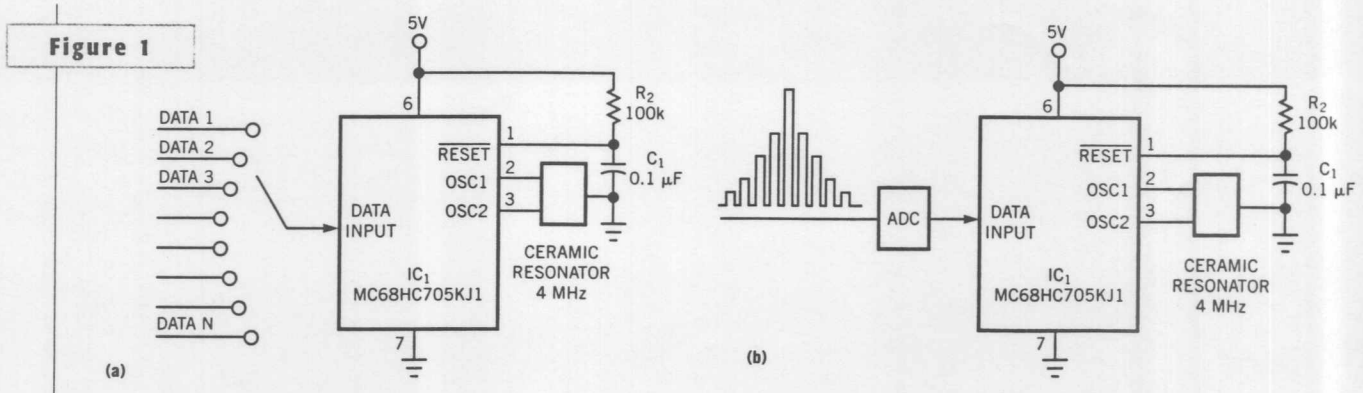
Microcontroller selects minimum/maximum value

Abel Raynus, Armatron International, Melrose, MA

MICROCONTROLLER-BASED SYSTEMS for measurement, sensor-data processing, or control, sometimes require you to determine a maximum or minimum data value. For example, in an object-detection system, such as a radar or sonar system, the microcontroller receives echo signals from multiple targets and then must select the closest one; in other words, it must determine the minimum distance. Another example is an automatic-tuning system in which the microcontroller acquires data and must determine the maximum or minimum values of the data. A simple way to determine maximum and minimum value involves a microcontroller, IC_1, that receives a set of data from N sources (**Figure 1**). The data should be in 8-bit format. To simplify the process, consider that the data are 1-byte-long integers. In other words, the data cover the range 0 to 255. In most cases, this range produces satisfactory results. The same limitation applies to the number of data sources. If you need more data precision or more data sources, then you can use two or more bytes at the expense of added program complication.

Two approaches exist for the maximum/minimum values. In the first, the microcontroller memory collects the received data and processes the data to de-

Figure 1



In one approach to determining minimum and maximum values, the microcontroller stores data values in memory before processing them (a); in another approach, it processes data on the fly (b).

R_8, sets the gain of power amplifier IC_1. Analog multiplexer IC_2 selects one of the eight tap voltages and applies it to IC_1's inverting feedback node. Because only one bias current flows through the nonlinear switches, the topology introduces no measurable distortion.

The amplifier changes gain in 3-dB steps, starting from a high of 6 dB, then decreasing to 3 dB, 0 dB,...−12 dB, and finally −∞ dB (in other words, "mute" or "off") as you step the volume bits V_2 to V_0 from 7 to 0. If you don't want to turn off the amplifier at the minimum-volume setting, you could change resistor R_9 to a finite value for whatever end-attenuation level you desire. The assumption in **Figure 1** is that your system has a way to generate the 3-bit volume codes. For those applications that do not have these bits available, you can use the circuit in **Figure 2** to generate them. Pushing switches S_1 and S_2 clocks up/down counter IC_3 up and down, respectively. Transistors Q_1 and Q_2 decode the counter's zero state and disable the down clock when the count reaches zero. In this fashion, the circuit limits the counter to a value of 0 to 7. Once you attain a maximum or minimum volume, further pushes on the up and down switches, respectively, have no effect on the volume setting until you go in the opposite volume direction. Standby-current consumption of the logic is almost entirely a function of the resistors you use for Q_1 to Q_3, because the logic chips use almost no power. You can set a nonzero power-up volume by using IC_3's load (LD) pin when you first apply power. In **Figure 2**, the counter powers up at a "4" volume, rather than "0" (muted). 74HCXX logic operates over 2 to 7V, but the power amplifier, IC_1, uses 2.7 to 5.5V. Therefore, IC_1 sets the operating-voltage range.

Is this the best Design Idea in this issue? Select at www.ednmag.com.

Tricks improve on Excel LCD initialization

Aubrey Kagan, Weidmüller Ltd, Markham, ON, Canada

A previous Design Idea on using Excel for LCD initialization relies on the user for cutting and pasting from Excel into an editor (**Reference 1**). It appears that it would take a minimum of six keystrokes or mouse strokes from one character to a new character to perform this operation. This procedure does not allow for the addition of comments, which would be advantageous in the maintenance of the character set. Over a set of 64 characters, a user would take nearly 400 actions. The use of macros would allow for the reduction of these keystrokes or mouse strokes and would probably improve the previous idea.

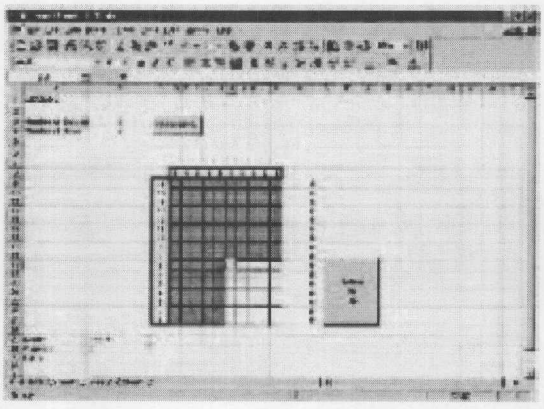


Figure 1 This Excel configuration simplifies LCD-matrix initialization.

In using such macros, rather than copy the numbers associated with characters into the text editor, as the previous Design Idea suggests, you append each sequence of numbers to a text file. When the character set is complete, you can open the text file, massage it with find-and-replace techniques, and then paste it as a whole into a target file. An added benefit is that each sequence appears on a single line rather than having a line for every row of the display. You can maintain this original file for use in both high-level and assembly programs if the need for repeating the process ever arises.

The macro in **Listing 1** performs the following tasks:

- Prompts the user for a comment associated with this character.
- Creates a line consisting of a leader, data separated by a delimiter, a comment symbol, and the comment already entered.
- Takes the file name from the spreadsheet. If the file does not exist, the macro creates a file with the line appended to it. Each time the macro runs, it opens, adds to, and closes the file.
- Clears the entries in the LCD matrix and sets the top-left cell of the matrix as active, ready for the next character.

All the parameters represent data in the cells of the spreadsheet. The file name is in cell A1. (The macro appends the ".txt.") Cell B26 has the leader, B27 has the comment symbol, and B28 has the delimiter between the numbers in the file. You can use this data in assembler or C or some other language, so the **listing** generalizes. C would ignore the "/* db */," and you could universally replace it with "db" if you use an assembler. Similarly, you could globally modi-

fy the comment notation, " ;//" (with a space before the semicolon), in the text file as well. The trailing comma of the line of data is needed in C as part of a constant array but in assembler would generate an error. If you're using an assembler, the global search would look for the sequence " ;//" and replace it with a space and a semicolon. Because of the inconsistent way text editors deal with tabs, the **listing** avoids using the tab character. By modifying the cells, you can easily customize the format. You can implement other changes by modifying the macro, written in Visual Basic for Applications. The following is a sample line from a file:

/* db */ 4 , 12 , 4 , 4 , 4 , 4 , 14 ,;//this is a "1" on a 5×7 matrix.

A button on the spreadsheet allows you to run the macro with a single click (**Figure 1**), so you need not switch from the mouse to the keyboard to run the macro. You can also run the macros from Ctrl-key combinations. (In Excel, you should note that, to run a macro, you must have completed data entry into a cell.) The matrix's 10×16-cell format is also generalized. You can alter this format by entering the number of columns in B3 and the number of rows in B4 and clicking on the Shade Matrix button. This action activates a second macro that shades out the cells not in use. Clicking on the Update to File button triggers a pop-up window requesting a comment. You need not enter the comment symbol, because the macro automatically enters it. The macro then appends the line to the file. The unused rows do not appear in the file. You can download the Excel file in **Listing 1** from the Web version of this Design Idea at www.endmag.com.

REFERENCE

1. Bitti, Alberto, "Excel offers painless LCD initialization" (*EDN*, Sept 20, 2001, pg 98).

Is this the best Design Idea in this issue? Select at www.ednmag.com.

LISTING 1—SAVE-CHARACTER EXCEL MACRO

```
Sub SaveCharacter()
'
' SaveCharacter Macro
' Macro recorded 11/16/01 by Aubrey Kagan
'
' Keyboard Shortcut: Ctrl+Shift+T
'
'if file does not exist then create it.
    Dim sFname As String
    Dim sX As String
    Dim iPoint As Integer
    Dim sComment As String
    Dim iUtil As Integer
    Dim iFileExists As Boolean

    sFname = Range("A1").Value
    sFname = sFname & ".txt"

    sX = Dir(sFname)
    If sX <> "" Then
        iFileExists = True
    Else
        iFileExists = False
        Open sFname For Output As #1
        'take the comment symbols an place the file name as the first
        'comment

        sX = Range("B27").Value & sFname
        Print #1, sX
        Close #1
        'file created and saved
    End If
    'now we print a range of values across the page
    'associated with the bytes for the pixels
    iPoint = 24
    iPoint = iPoint - (Range("B4").Value - 1)
    sX = iPoint
    'converting typ to string
    sX = "R" & sX
    sX = sX & "C14"
    'pointing at the start of the data
    Application.Goto Reference:=sX

    'get character description
        sComment = InputBox("Enter you comment for this Character", "Comment
Creation")

        Open sFname For Append As #1
        Print #1, Range("B26").Value;

        For iUtil = 0 To (Range("B4").Value - 1)
            sX = (iPoint + iUtil)
            sX = "R" & sX
            sX = sX & "C14"
            'placing the cursor at the data address
            Application.Goto Reference:=sX
            Print #1, ActiveCell.Value;
            Print #1, Range("B28").Value;
            'print value followed by the delimiter
        Next

        sComment = Range("B27").Value & sComment
        Print #1, sComment

        'ensure all files are closed
        Close
    'as a final step clear the page for the next character
        Range("D9:M24").Select
        Selection.ClearContents
        'move to the top left hand of screen
        'first deal with rows
        iPoint = 24
        iPoint = iPoint - (Range("B4").Value - 1)
        sX = iPoint
        'type conversion
        sX = "R" & sX & "C"
        'concantenate for rows and add C for columns
        iPoint = 13
        iPoint = iPoint - (Range("B3").Value - 1)
        sX = sX & iPoint
        'should do the conversion as well
        Application.Goto Reference:=sX

    End Sub
```